Lookahead Actions in Dispatching to Parallel Queues

Esa Hyytiä

Department of Communications and Networking, Aalto University, Finland

Abstract

Applying the first policy iteration (FPI) to any static dispatching (task assignment) policy yields a new improved dynamic policy that takes into account the particular cost structure and the expected future arrivals. However, it is generally hard to go beyond that due to the complex state space and the resulting difficulty in computing the value function for a dynamic policy. For example, applying FPI to identical FCFS servers with Bernoulli split gives the Least-Work-Left (LWL) policy, for which no closed-form value function is known. In fact, LWL with identical servers is equivalent to an M/G/k queue, the performance measures of which have remained as open problems. The situation gets even more complicated with heterogeneous servers. In this paper, we take an intermediate approach and consider *lookahead actions* that concern not only the current job but also the job arriving next, after which a basic (static) policy is assumed to take over. This is important as the benefits from some decisions can only be reaped with appropriate subsequent actions. The lookahead enables sound estimates also for marginal admission costs, e.g., with respect to LWL. The superior performance of the new near-optimal dispatching policies is demonstrated numerically.

Keywords: Lookahead, Policy improvement, MDP, Task assignment, Decision tree

1. Introduction

For a system of parallel servers, one fundamental resource allocation question is the so-called dispatching problem (i.e., the task assignment or routing problem), where the task is to choose a server for each arriving job so as to minimize the mean waiting time, energy consumption, or some other performance metric of interest. This general problem has many applications within ICT such as routing in data networks (where different links are seen as servers) and task assignment in web server farms, distributed parallel computing, (mobile) cloud computing and data centers. The dispatching problem arises also in many other contexts such as car toll stations in a highway, cashiers in a supermarket, manufacturing systems, etc.

In the basic setting, all tasks (i.e., jobs or customers) are subject to assignment, i.e., all tasks are *flexible* [1]. In general, servers could also receive dedicated inflexible tasks that cannot be served elsewhere due to technical or policy reasons. The so-called cycle stealing is a related concept where only the tasks of the beneficiary system can be re-routed to a donor system given that the latter is idle [2, 3]. We assume flexible customers, but note that inflexible customers can be introduced without any technical difficulty. The optimal dispatching policy depends on the objective (cost function), and on the available information about the jobs and the state of the system (service times, number of jobs in each queue, etc.). Jobs arriving in future are typically assumed to constitute a Poisson process, which is also our assumption. Moreover, we assume a size-aware setting where the service time of a job becomes known upon arrival. For a more detailed overview of task assignment problems and policies, we refer to the recent book [4].

Three approaches to solve a dispatching problem can be identified. First, one can pick a promising dispatching policy and just show its optimality. For example, Winston [5] has shown the optimality of the join-the-shortest-queue (JSQ) for identical first-come-first-served (FCFS) servers when the number in queue is available and exponentially distributed jobs arrive according to a Poisson process. Similarly, the round-robin (RR) has been shown to be the optimal policy for identical FCFS servers, when the queues were initially in the same state [6–8].

☆This work was supported by the Academy of Finland (grant no. 260014).
Email address: `esa.hyytia@aalto.fi` (Esa Hyytiä)

Second, one can choose a parameterized dispatching policy $\alpha(h)$, derive the mean cost rate for it, and choose the optimal h. For example, the Size-Interval-Task-Assignment (SITA) bases the dispatching decision solely on the size of the job [9, 10]: the range of job sizes is divided to non-overlapping intervals $[\xi_{i-1}, \xi_i)$ by the cutoff thresholds ξ_i and a job with size $x \in [\xi_{i-1}, \xi_i)$ is routed to Server i. The cutoff thresholds ξ_i are the parameters of SITA. As the dispatching decision does not depend on the state of the queues, SITA is a static (state-independent) policy and, assuming FCFS and Poisson arrivals, an expression for the mean waiting time follows from the Pollaczek-Khinchine mean value formula. SITA has been shown to be the optimal static size-aware policy for Poisson arrivals with respect to the mean waiting and sojourn time [11].

The third approach is based on the policy improvement of the Markov decision processes (MDP) [12–14]. Given the so-called value function, one carries out the first policy iteration (FPI) step, which typically yields the greatest improvement towards the optimal policy. In the context of dispatching problems, this approach has been utilized to minimize the blocking probability, see Krishnan [15, 16] and Leeuwaarden *et al.* [17], and the sojourn time (i.e., delay or latency) or its generalization by arbitrary holding costs, see, e.g., Krishnan [18], Sassen *et al.* [19], Bhulai *et al.* [20] and Hyytiä *et al.* [21–23]. Most dispatching systems considered have a rather complex state space (e.g., infinite number of waiting places, a continuous range of remaining service time, etc.) and therefore the application of the policy improvement has been limited to static policies, in case of which, the system decomposes and it is sufficient to analyze isolated single server queues. In this paper, we *dig* a bit deeper and take an intermediate approach, where also the next decision(s) can be *dynamic*, after which a static basic policy takes over. The decision for the next job is only tentative (i.e., a plan), and re-evaluated upon an actual arrival. There is a strong analogy to a computer program that builds a decision tree when playing a game, with the distinction that we have a continuous time dimension and a non-discrete state space. Therefore, the decision tree in our case is only two levels deep (lookahead of two), the leaves of which are evaluated assuming a static basic policy. Note that almost all commonly used policies are index-policies, where an index is computed independently for each queue and the queue with the smallest index is chosen. This includes the Least-Work-Left (LWL), JSQ, Bernoulli split, SITA and all FPI policies based on static basic policies. However, as we consider not only one but two (or more) subsequent decisions, the corresponding policy, referred to as the *Deep* (FPI) policy, *is not an index-policy*, but belongs to a more general family of dispatching policies.

The rest of the paper is organized as follows. Section 2 describes the system model, notation and earlier results for the basic FPI. In Section 3, we derive the marginal cost for the static, and in Section 4 for the dynamic second actions. Section 5 illustrates how these lookahead actions improve the performance due to the more "accurate" estimates for marginal costs, and Section 6 contains the conclusions.

2. Preliminaries

Suppose there is a system with k servers, with individual queues, that process incoming jobs according to the FCFS scheduling discipline with service rates ν_i, $i = 1, \ldots, k$. The job arrival process is a Poisson process with rate λ and it constitutes a sequence,

$$(X^{(1)}, B^{(1)}),\ (X^{(2)}, B^{(2)}),\ \ldots$$

where $X^{(j)}$ denotes the size and $B^{(j)}$ the *holding cost rate* (per unit time) of Job j. Each jobs incurs costs at that rate until it departs the system. The jobs are further assumed to be i.i.d., $(X^{(j)}, B^{(j)}) \sim (X, B)$. However, $B^{(j)}$ may still depend on the corresponding job size $X^{(j)}$. For example, the *slowdown* of Job j is defined as the ratio of the sojourn time $T^{(j)}$ to the job size $X^{(j)}$ [24]

$$\gamma = \frac{T^{(j)}}{X^{(j)}},$$

corresponding to the holding cost rates $B^{(j)} = 1/X^{(j)}$. With $B = 1$, the objective is to minimize the mean delay (i.e., latency or sojourn time). In a multi-class setting, each customer class may have a different holding cost rate.

We let α denote the dispatching policy, which can either be *static* (state-independent) or *dynamic* (state-dependent), where the decisions of the latter depend on the state of the queues. The decisions of a static policy may still depend on the size and the holding cost of a job. Examples of static policies are the Bernoulli split (RND), which assigns the arriving jobs independently with probabilities $p_1, \ldots, p_k$, and the already mentioned SITA. Assuming a Poisson arrival process, the static policies decompose the system into k independent queues, which facilitates the analysis.

The so-called *basic policy* α_0 is an arbitrary static policy. Let λ_i denote the job arrival rate to Queue i with α_0, and λ the total arrival rate, $\lambda = \lambda_1 + \ldots + \lambda_k$. Then let S_i denote the service time and B_i the holding cost of a random job that α_0 assigns to Queue i. If α_0 is RND, we have $\lambda_i = p_i\lambda$ and $S_i \sim X/\nu_i$. The mean service time of a random job with an arbitrary static α_0 is

$$\mathrm{E}[S] = \frac{1}{\lambda}\sum_i \lambda_i \mathrm{E}[S_i]. \tag{1}$$

The Pollaczek-Khinchine mean value formula gives the mean sojourn time $\mathrm{E}[T]$ with α_0,

$$\begin{aligned} \mathrm{E}[T] &= \frac{1}{\lambda}\sum_i \lambda_i \left(\mathrm{E}[W_i] + \mathrm{E}[S_i]\right) \\ &= \frac{1}{\lambda}\sum_i \lambda_i \left(\frac{\lambda_i \mathrm{E}[S_i^2]}{2(1-\rho_i)} + \mathrm{E}[S_i]\right), \end{aligned} \tag{2}$$

where $\rho_i = \lambda_i\, \mathrm{E}[S_i]$. In particular, the mean cost per job with α_0 is

$$\begin{aligned} \mathrm{E}[TB] &= \frac{1}{\lambda}\sum_i \lambda_i \mathrm{E}[T_i B_i] \\ &= \frac{1}{\lambda}\sum_i \lambda_i \left(\mathrm{E}[W_i]\, \mathrm{E}[B_i] + \mathrm{E}[S_i B_i]\right), \end{aligned} \tag{3}$$

where the latter is due to the fact that the waiting time is independent of the service time with FCFS.

2.1. *Value function of M/G/1-FCFS*

In general, a value function characterizes the difference in cumulative costs between a system that is initially in state $\mathbf{z}$ and a system that is initially in equilibrium,

$$v_{\mathbf{z}} \triangleq \lim_{t\to\infty} \mathrm{E}[C_{\mathbf{z}}(t) - rt],$$

where $C_{\mathbf{z}}(t)$ denotes the cumulative costs during $(0, t)$ when initially in state $\mathbf{z}$, and r is the mean cost rate [12–14]. The system is assumed to be ergodic and stable, so that the above limit is well-defined. In the MDP framework, the value functions are used for the policy improvement as they enable a direct comparison of two initial states $\mathbf{z}_1$ and $\mathbf{z}_2$ by $v_{\mathbf{z}_1} - v_{\mathbf{z}_2}$. A constant offset in $v_{\mathbf{z}}$ is immaterial and it is sufficient to know the relative value $v_{\mathbf{z}} - v_0$, where 0 is some specific reference state.

In our context, a static basic policy α_0 feeds a Poisson process to each queue and the system decomposes to k independent M/G/1-FCFS queues. The value function of a size-aware M/G/1-FCFS queue is quadratic [22], and for Queue i we have

$$v_{u_i}^{(i)} - v_0^{(i)} = \frac{\lambda_i\, \mathrm{E}[B_i]}{2(1-\rho_i)}(u_i)^2, \tag{4}$$

where λ_i, ρ_i and u_i are the arrival rate, offered load and backlog in Queue i. The value function $v_{u_i}^{(i)}$ describes how much more costs the *future* arrivals on average incur in Queue i than they would if the system was initially in equilibrium. Due to the decomposition, the value function of the whole system is the sum of the queue-specific value functions

$$v_{\mathbf{z}} = \sum_{i=1}^{k} v_{u_i}^{(i)},$$

where $\mathbf{z} = (u_1, \ldots, u_k)$ denotes the system state. Let g_i denote the constant factor in the value function,

$$g_i = \frac{\lambda_i\, \mathrm{E}[B_i]}{2(1-\rho_i)},$$

so that $v_{u_i}^{(i)} = g_i u_i^2 + v_0^{(i)}$.

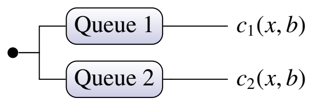


Figure 1: Decision tree corresponding to the basic FPI, which deviates from a static α_0 only for the first decision. The size x and holding cost b of the current job are known.

Suppose next that a new job arrives and induces a state change. First we note that (4) includes only the costs related to the future arrivals. The so-called immediate cost is associated with the state change, which in our case is the costs that the new job will incur. According to FPI, the *marginal cost* of assigning a job with size x and holding cost b to Queue i is

$$\begin{aligned} c_i(x, b) &\triangleq (u'_i + x/\nu_i)b + v_{\mathbf{z}} - v_{\mathbf{z}'} \\ &= u'_i b + (x/\nu_i)(b + g_i(2u'_i + x/\nu_i)), \end{aligned}$$

where u'_i denotes the *a priori* backlog in Queue i, and $\mathbf{z}'$ and $\mathbf{z}$ denote the state of the system before and after the inclusion of the new job. The term $(u'_i + x/\nu_i)b$ is the immediate cost (sojourn time of the current job times its holding cost), and $v_{\mathbf{z}} - v_{\mathbf{z}'}$ is the penalty the future arrivals experience. That is, the marginal cost is the expected increase in the infinite time-horizon costs for admitting a job to Queue i instead of rejecting it. Note that $c_i(x, b)$ depends on the backlog in Queue i, but not on the state of the other queues. Alternatively, using the *a posteriori* backlogs gives

$$c_i(x, b) = u_i b + g_i(x/\nu_i)(2u_i - x/\nu_i), \tag{5}$$

where u_i includes the new job, $u_i = u'_i + x/\nu_i$. The marginal cost $c_i(x, b)$ describes how much more jobs collectively incur costs in the system if the new job with size x and holding cost b is assigned to Queue i (given the subsequent decisions are according to α_0). The improved policy α_{FPI} chooses the queue with the smallest cost, $\alpha_{\text{FPI}} = \operatorname{argmin}_i c_i(x, b)$. The decision tree in Fig. 1 illustrates this for a small system of two queues. As $c_i(x, b)$ depends only on the (local) state of the Queue i, each queue can compute the cost independently. The resulting policy is thus an index-policy, where the queue with the smallest index is chosen.

Example

Consider $k = 2$ identical servers ($\nu_i = 1$), a constant holding cost $B = 1$, and (uniform) Bernoulli split such that $\lambda_i = \lambda/2$ and $\rho_i = \rho/2$. The value function of the whole system is

$$v(u_1, u_2) - v(0, 0) = \frac{\lambda}{2(2 - \rho)}(u_1^2 + u_2^2),$$

where $\rho < 2$ is assumed for stability. The marginal cost of Queue i in this case is $c_i(x) = u_i + xg(2u_i - x)$, and the FPI step yields the LWL policy that assigns the new job to the queue with the shortest backlog. This policy is also greedy as it minimizes the sojourn time of the new job. We know that LWL is not generally optimal with respect to delay, and the standard approach to get further would be to carry out the second policy iteration step. That would require determining the value function for LWL, which is a non-trivial task.

3. Static second action

We choose an intermediate approach. After the initial decision, instead of immediately returning back to the basic policy α_0, we let another policy α_1 make the next decision, after which the basic policy takes over. We consider two cases: static and dynamic second decision, where the dynamic second action means that the decision of α_1 depends on the arrival time of the next job. Interestingly, it is possible that a dynamic policy makes a static second action. For example, LWL will choose the queue which currently has the least amount of work independently of the size, holding cost, or arrival time of the job. In our case, we know the current backlogs and thus the queue for the *next* job is already known. Hence, the second action by LWL is static[1] (i.e., known in advance, whereas the third action is not). Similarly, the Round-robin policy is deterministic and the second (or any other) action by it is static.

[1]This holds when ties are broken appropriately, e.g., when queues become idle.

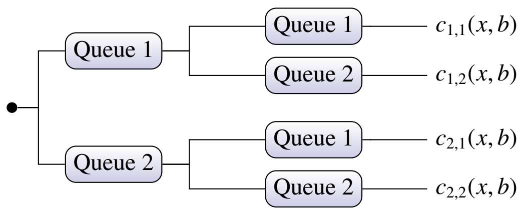


Figure 2: Decision tree corresponding to Deep (FPI), which deviates from a static α_0 for two decisions, where the size, holding cost and arrival time of the second job are unknown. The latter decision is tentative.

In this section, we consider such policies α_1 whose decisions for the next job are static. The dynamic second actions are discussed in Section 4. We start by determining the marginal cost $c_{i_0,i_1}(x,b)$ associated with a fixed action (i_0,i_1) that assigns the current new job with size x and holding cost b to Queue i_0, the next job to Queue i_1, and the subsequent jobs according to α_0. The corresponding decision tree is illustrated in Fig. 2 for $k=2$ servers. Whereas the basic FPI considers k alternatives, iterating over the next decision increases the number of combinations to k^2. By conditioning on the size and holding cost of the next job, we obtain the marginal cost also for an arbitrary α_1 that is static in the sense that the next decision may depend on the size and/or the holding cost, but not on the arrival time.

3.1. Fixed second action

Next we determine the marginal cost of a fixed action (i_0,i_1) assigning the current job to Queue i_0 and the next job (tentatively) to Queue i_1. Let the u'_i denote the *a priori* and u_i the *a posteriori* backlogs,

$$u_i = \begin{cases} u'_i, & i \neq i_0 \\ u'_i + x/\nu_i & i = i_0, \end{cases}$$

and the U_i the backlogs upon the arrival of the next job,

$$U_i \triangleq (u_i - \tau)^+,$$

where τ denotes the time to the next arrival, $\tau \sim \mathrm{Exp}(\lambda)$.

Proposition 1 *The marginal cost of a fixed action (i_0,i_1) assigning the current job with size x and holding cost b to Queue i_0, the next job to Queue i_1, and the subsequent jobs with a static α_0 is*

$$\begin{aligned} c_{i_0,i_1}(x,b) = u_{i_0} b + g_{i_0}\frac{x}{\nu_{i_0}}\left(2u_{i_0} - \frac{x}{\nu_{i_0}}\right) + \frac{2}{\lambda^2}\sum_i g_i(1-\lambda u_i - e^{-\lambda u_i}) - \mathrm{E}[TB] \\ + \left(\mathrm{E}[B] + \frac{2g_{i_1}\mathrm{E}[X]}{\nu_{i_1}}\right)\left(u_{i_1} - \frac{1-e^{-\lambda u_{i_1}}}{\lambda}\right) + g_{i_1}\frac{\mathrm{E}[X^2]}{\nu_{i_1}^2} + \frac{\mathrm{E}[XB]}{\nu_{i_1}}. \end{aligned} \tag{6}$$

Proof: Let w_0 denote the (known) waiting time of the new job, $w_0 = u'_{i_0}$, and W_1 the waiting time of the next job, $W_1 = U_{i_1}$. The marginal cost of action (i_0,i_1) is

$$c_{i_0,i_1}(x,b) = (w_0 + x/\nu_{i_0})b + \mathrm{E}[(W_1 + X_1/\nu_{i_1})B_1 - r\,\tau] + \mathrm{E}[v^{(i_1)}_{U_{i_1}+X_1/\nu_{i_1}} + \sum_{i\neq i_1} v^{(i)}_{U_i}] - v_{\mathbf{z}'},$$

where X_1 and B_1 denote the size and holding cost of the next job, and r is the mean rate at which the system incurs costs, $r = \lambda\,\mathrm{E}[TB]$, where $\mathrm{E}[TB]$ is according to α_0 and given by (3). The first two terms correspond to the relative costs incurred until and including the next arrival, and the remaining terms take care of the later arrivals that are assigned according to the basic policy α_0. As $(X_1,B_1)\sim(X,B)$, substituting (4) into above gives

$$c_{i_0,i_1}(x,b) = u_{i_0}b + \mathrm{E}[U_{i_1}]\mathrm{E}[B] + \mathrm{E}[XB]/\nu_{i_1} + g_{i_1}\mathrm{E}[(U_{i_1} + X/\nu_{i_1})^2] + \sum_{i\neq i_1} g_i \mathrm{E}[(U_i)^2] - \mathrm{E}[TB] - \sum_i g_i (u'_i)^2,$$

where, as explained earlier, B may depend on X (of the same job), while $\mathrm{E}[U_{i_1}B] = \mathrm{E}[U_{i_1}]\mathrm{E}[B]$ due to their independence. As U_{i_1} and X are also independent, we have

$$\mathrm{E}[(U_{i_1} + X/\nu_{i_1})^2] = \mathrm{E}[U_{i_1}^2] + 2\frac{\mathrm{E}[X]}{\nu_{i_1}}\,\mathrm{E}[U_{i_1}] + \frac{\mathrm{E}[X^2]}{\nu_{i_1}^2}.$$

Hence,

$$c_{i_0,i_1}(x,b) = u_{i_0}b - \mathrm{E}[TB] + \sum_i g_i(\mathrm{E}[U_i^2] - (u_i')^2) + \left(\mathrm{E}[B] + \frac{2g_{i_1}\,\mathrm{E}[X]}{\nu_{i_1}}\right)\mathrm{E}[U_{i_1}] + g_{i_1}\frac{\mathrm{E}[X^2]}{\nu_{i_1}^2} + \frac{\mathrm{E}[XB]}{\nu_{i_1}}. \tag{7}$$

For $\mathrm{E}[U_i]$, we obtain

$$\mathrm{E}[U_i] = \int_0^\infty \lambda e^{-\lambda t}(u_i - t)^+\,dt = u_i - (1 - e^{-\lambda u_i})/\lambda,$$

which gives the mean waiting time of the next job, $\mathrm{E}[W_{i_1}] = \mathrm{E}[U_{i_1}]$. Similarly, for $\mathrm{E}[U_i^2]$ we have

$$\mathrm{E}[U_i^2] = \int_0^{u_i} \lambda e^{-\lambda t}(u_i - t)^2\,dt = u_i^2 + \frac{2(1 - e^{-\lambda u_i} - \lambda u_i)}{\lambda^2}.$$

Substituting these back into (7) gives (6). □

We observe that the first two terms in (6) correspond to (5) and the remaining terms to the marginal cost of fixing also the next queue instead of returning immediately back to α_0. We recall that (5) depends only on the state of the Queue i. In contrast, (6) depends on the state of every queue (i.e., not just Queues i_0 and i_1) and the corresponding policy *is not* an index-policy. Note also that if α_0 is a Bernoulli split, then

$$g_{i_1}\frac{\mathrm{E}[X^2]}{\nu_{i_1}^2} = \mathrm{E}[W_{i_1}],$$

i.e., the mean waiting time in Queue i_1 with α_0, and

$$\frac{\mathrm{E}[XB]}{\nu_{i_1}}$$

is the mean service cost in Queue i_1 with α_0. Moreover, with $B = 1$, their sum corresponds to the mean delay in Queue i_1 with α_0. As dispatching the next job to Queue i_1 was an arbitrary choice, we can relax that and iterate over it

$$\begin{aligned}\tilde{c}_{i_0}(x,b) = {} & u_{i_0}b + g_{i_0}\frac{x}{\nu_{i_0}}\left(2u_{i_0} - \frac{x}{\nu_{i_0}}\right) + \frac{2}{\lambda^2}\sum_i g_i(1 - \lambda u_i - e^{-\lambda u_i}) - \mathrm{E}[TB] \\ & + \min_j\left\{\left(\mathrm{E}[B] + \frac{2g_j\,\mathrm{E}[X]}{\nu_j}\right)\left(u_j - \frac{1 - e^{-\lambda u_j}}{\lambda}\right) + g_j\frac{\mathrm{E}[X^2]}{\nu_j^2} + \frac{\mathrm{E}[XB]}{\nu_j}\right\}.\end{aligned} \tag{8}$$

Referring to Fig. 2, this means choosing the optimal branch at the second level of the decision tree.

Lemma 2 *For identical servers ($\nu_i = \nu_j$) and uniform RND as the basic policy ($g_i = g_j$),* (8) *ends up "assigning" the next job according to LWL, i.e., $i_1 = \operatorname{argmin}_i u_i$.*

Proof: Follows from noting that

$$\left(u_j - \frac{1 - e^{-\lambda u_j}}{\lambda}\right),$$

is a strictly increasing function of u_j for $u_j > 0$. □

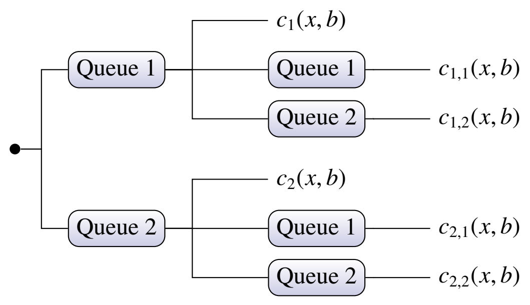


Figure 3: Decision tree combining the fixed second actions and the immediate return to α_0.

Instead of (tentatively) fixing the second action to Queue i_1, we can also decide to return to α_0 immediately. The marginal cost of this action, $c_{i_0}(x, b)$, is given by (5). In this case, we have $k + 1$ possible second actions. The corresponding decision tree is illustrated in Fig. 3. The optimal second action depends on the current state, the basic policy and the new job, and each of the $k+1$ choices can turn out to yield the lowest cost. The corresponding marginal cost for Queue i_0 with the optimal (tentative) second action is

$$c^*_{i_0}(x, b) = \min\{\min_j c_{i_0,j}(x, b), c_{i_0}(x, b)\}. \tag{9}$$

Lemma 3 *$c_{i_0}(x, b) \geq \min_j c_{i_0,j}(x, b)$ for any α_0 that is a Bernoulli split (RND).*

Proof: With RND, $c_{i_0}(x, b) = \sum_j p_j c_{i_0,j}(x, b)$. □

Note that for SITA the above does not hold, and in some cases $c_{i_0}(x, b)$ turns out to give a lower marginal cost than what fixing the second action can give. This suggests that the particular basic policy is actually doing a relatively good job in such cases. The more general static second actions, discussed in the next section, can incorporate desirable features of such α_0 directly into the second action.

3.2. General static second action

Until now, we have assumed that the next job is assigned to Queue i_1, where i_1 is either fixed (e.g., according to LWL) or the one yielding the smallest marginal cost. However, it is straightforward to show that i_1 can be also chosen by an arbitrary static policy, which includes, e.g., job size and holding cost based decisions (e.g., in SITA fashion, a short job to Queue i_0, and otherwise to some other queue). The corresponding marginal cost is obtained by conditioning on the different arrival types, each of which has a predetermined queue for the next arrival. We omit details for brevity.

Example

Consider two identical parallel queues with unit service rates $\nu_i = 1$, constant holding cost $B = 1$, exponentially distributed service times, $X \sim \text{Exp}(1)$, arrival rate $\lambda = 1.5$ and uniform Bernoulli split as the basic policy, α_0=RND. According to Lemmas 2 and 3, the second decision by LWL yields always a lower cost than choosing the other queue or returning back to α_0 immediately. Fig. 4 illustrates the decision by Static Deep (with α_1=LWL) for different job sizes x in (u'_1, u'_2)-plane (the *a priori* backlogs). We observe that Static Deep is not a *switch-over* policy at $x = 0.5$.

Consider next the following four actions: (i) the current job to Queue 1, (ii) the current job to Queue 2, (iii) the current job to Queue 1 and the next according to LWL, (iv) the current job to Queue 2 and the next according to LWL. In every case, the subsequent jobs are assigned according to α_0. Thus, the first two actions correspond to the basic FPI approach, and the next two to Deep with α_1=LWL. Fig. 5 illustrates the marginal costs for fixed $u'_1 = 2$ and $x = 0.5$. The x-axis corresponds to the backlog in Queue 2, u'_2, and the y-axis to the marginal cost relative to action (i). The basic FPI, choosing the action (i) or (ii), reduces to LWL with a switch-over point at $\xi_{\text{FPI}} = 2$ (i.e., Queue 2 is chosen when $u'_2 > \xi_{\text{FPI}}$). We observe that it is indeed always beneficial to fix a queue also for the next arrival (even though the arrival and service times of the next job are currently unknown), in accordance with Lemma 3. We already observed in Fig. 4 that Static Deep is not a switch-over policy. This can be seen also in Fig. 5, where the curves corresponding to actions (iii) and (iv) cross each other not just once but three times.

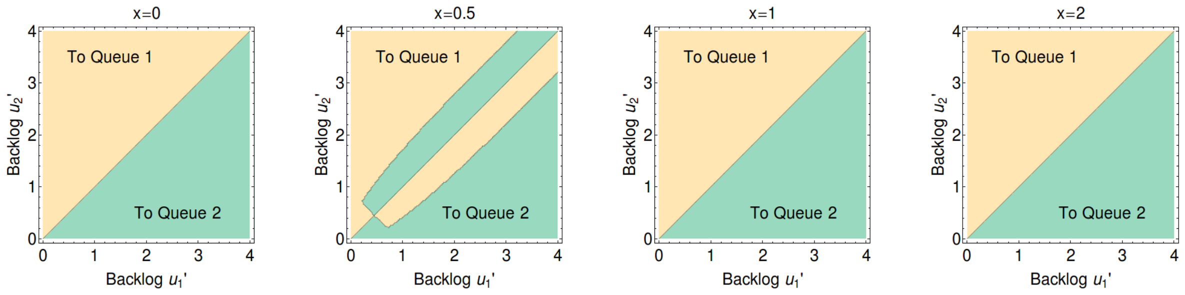


Figure 4: Decisions with Static Deep when α_1=LWL and α_0=RND for different job sizes x. Example system has two identical servers $\nu_1 = \nu_2 = 1$, exponentially distributed job sizes $X \sim \text{Exp}(1)$, unit holding cost $B = 1$ and arrival rate $\lambda = 1.5$. The Deep policy is not switch-over type at $x = 0.5$.

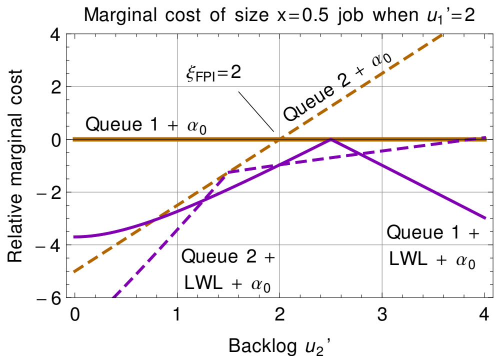


Figure 5: Expected marginal costs with four different actions. In this case, it is always advantageous to fix also the next decision according to LWL.

4. Dynamic second action

Let us consider next the dynamic second actions, which give us even more flexibility in defining how the next job is treated. By dynamic, we mean that the action depends also on the arrival time, i.e., it adapts to the changes in the state of the system. This is especially useful when the service rates ν_i are heterogeneous, and choosing the queue with the shortest backlog (i.e., LWL) may not be the same as choosing the queue that minimizes the sojourn time of the job. The latter, referred to as the Myopic policy, chooses the queue according to the *posteriori* backlogs

$$\underset{i}{\text{argmin}}\; u_i' + x/\nu_i,$$

and thus it minimizes the sojourn time and also the holding costs of the new job. LWL and Myopic are different only when $\nu_i \neq \nu_j$ for some servers i and j. It is also easy to see that, unlike LWL, Myopic is not a static policy with respect to the next arrival. Suppose that $u_1 < u_2 < u_3$ and $\nu_1 < \nu_2 < \nu_3$, i.e., the slowest server has the shortest backlog (measured in time). In this case, Myopic assigns very short jobs to Queue 1, and very long jobs to Queue 3, etc. However, as time passes without new arrivals, first Queue 1, then Queue 2, and eventually also Queue 3, all become empty and Myopic will assign the next job irrespectively of its size to the fastest Queue 3.

Note that with respect to the next arrival, we can neglect Queue i if $u_i \geq u_j$ and $\nu_i \leq \nu_j$ for some $j \neq i$, as then Queue i is never a better option than Queue j for the next job. In particular, if the fastest queue has currently the shortest backlog, then the next job will be unconditionally assigned to it. Thus, we can renumber the queues so that for the first k^* queues it holds that $u_1 < \ldots < u_{k^*}$ and $\nu_1 < \ldots < \nu_{k^*}$, and the possible other queues we can neglect when considering the next arriving job. This state- and size-dependent myopic action is illustrated in Fig. 6 and the thresholds $x_i(t)$ are

$$x_i(t) = \frac{\nu_i \nu_{i+1}}{\nu_{i+1} - \nu_i}(u_{i+1} - \max\{t, u_i\})^+, \qquad i = 1, \ldots, k^*. \tag{10}$$

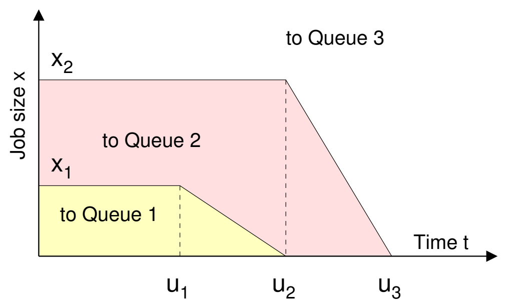


Figure 6: The action by Myopic minimizing the sojourn time of the next job (and thus also the holding costs) depends on the arrival time t and the size of the job x, but not on the holding cost b.

Hence, each threshold $x_i(t)$ remains first constant until Queue i becomes empty, and then the threshold decreases linearly to zero with slope $\nu_i \nu_{i+1}/(\nu_{i+1} - \nu_i)$, where $\nu_i < \nu_{i+1}$.

4.1. General case

Consider first the general case and let $I_i(t)$ denote the set of jobs that an intermediate policy α_1 assigns to Queue i on condition that the next job arrives at time t. Similarly as with the static case, we need to determine the marginal cost of action (i_0, α_1) that chooses Queue i_0 for the current job, a queue with α_1 for the next job, after which the basic policy α_0 takes over. Let $(S_i(t), B_i(t))$ denote the conditional service time and holding cost of a job that α_1 assigns to Queue i at time t,

$$\begin{aligned} S_i(t) &\sim (X/\nu_i \,|\, (X,B) \in I_i(t)), \\ B_i(t) &\sim (B \,|\, (X,B) \in I_i(t)). \end{aligned}$$

Note that $S_i(t)$ and $B_i(t)$ may depend on each other as X and B may have dependence (cf. the slowdown metric). Both $S_i(t)$ and $B_i(t)$ also depend on $\mathbf{z}$ and i_0, which we have omitted from the notation for clarity.

Proposition 4 *The marginal cost of a dynamic action (i_0, α_1) assigning the current job of size x and holding cost b to Queue i_0, the next job with a dynamic α_1, and the subsequent jobs with a static α_0 is*

$$\begin{aligned} c_{i_0,\alpha_1}(x,b) &= u_{i_0} b + g_{i_0} \frac{x}{\nu_{i_0}} \left(2u_{i_0} - \frac{x}{\nu_{i_0}} \right) + \frac{2}{\lambda^2} \sum_i g_i (1 - \lambda u_i - e^{-\lambda u_i}) - \mathrm{E}[TB] \\ &+ \sum_{i=1}^{k} \int_0^\infty \lambda e^{-\lambda t}\, \mathrm{P}\{(X,B) \in I_i(t)\} \Big((u_i - t)^+ (\mathrm{E}[B_i(t)] + 2g_i \mathrm{E}[S_i(t)]) + g_i \mathrm{E}[S_i(t)^2] + \mathrm{E}[S_i(t) B_i(t)] \Big)\, dt. \end{aligned} \tag{11}$$

Proof: Conditioning on the arrival time t and the decision sets $I_i(t)$ gives

$$\begin{aligned} c_{i_0,\alpha_1}(x,b) &= u_{i_0} b + \int_0^\infty \lambda e^{-\lambda t} \Big[\sum_{i=1}^{k} \mathrm{P}\{(X,B) \in I_i(t)\} \Big(\mathrm{E}[((u_i - t)^+ + S_i(t)) B_i(t)] \\ &\quad - rt + g_i \mathrm{E}[((u_i - t)^+ + S_i(t))^2] + \sum_{j \neq i} g_j ((u_j - t)^+)^2 \Big) \Big] dt - \sum_i g_i u_i'^2 \\ &= u_{i_0} b - \mathrm{E}[TB] + \sum_i g_i \int_0^{u_i} \lambda e^{-\lambda t} (u_i - t)^2\, dt - \sum_i g_i u_i'^2 + \\ &\quad + \sum_{i=1}^{k} \int_0^\infty \lambda e^{-\lambda t}\, \mathrm{P}\{(X,B) \in I_i(t)\} \Big((u_i - t)^+ (\mathrm{E}[B_i(t)] + 2g_i \mathrm{E}[S_i(t)]) + g_i \mathrm{E}[S_i(t)^2] + \mathrm{E}[S_i(t) B_i(t)] \Big)\, dt. \end{aligned}$$

The integral on the first row we already know

$$\sum_i g_i \int_0^{u_i} \lambda e^{-\lambda t}(u_i - t)^2 \, dt = \sum_i g_i \mathrm{E}[U_i^2],$$

which gives (11). $\square$

The first two terms in (11) correspond to the marginal cost of i_0, the third term depends only on the first action i_0 but not on α_1, and the last summation depends on both. In particular, the first row is the same as in (6) and (8), and thus the other terms correspond to the difference in the marginal costs between these actions. Note also that sets $I_i(t)$ are independent of the arrival time t with any static second action α_1, and (11) reduces to what was discussed in Section 3. Similarly, also (11) depends on the state of each queue and the corresponding policy *is not* an index-policy.

4.2. Myopic α_1

Let us consider next the important special case when the second action is according to the Myopic policy, α_1 = Myopic. In this case, the sets $I_i(t)$ are as depicted in Fig. 6, and the integral in (11) can be developed further. Suppose that the backlog and service rates are as in the figure. Consider first Queue 1 having the shortest backlog. The threshold $x_1(t)$, given in (10), is initially constant and as $t = u_1 \ldots u_2$, $x_1(t)$ decreases linearly to zero and remains there. Hence, the integral in (11) for $i = 1$ reduces to

$$\begin{aligned} F(x_1(0))\Bigg[\left(u_1 - \frac{1-e^{-\lambda u_1}}{\lambda}\right)(\mathrm{E}[B_1(0)] + 2g_1\mathrm{E}[S_1(0)]) + (1-e^{-\lambda u_1})(g_1\mathrm{E}[S_1(0)^2] + \mathrm{E}[S_1(0)B_1(0)])\Bigg] \\ + \int_{u_1}^{u_2} \lambda e^{-\lambda t} F(x_1(t))\,(g_1\mathrm{E}[S_1(t)^2] + \mathrm{E}[S_1(t)B_1(t)])\, dt, \end{aligned}$$

where $F(\cdot)$ is the CDF of the job size distribution.

Similarly it is easy to compute the corresponding integral also for $i \neq 1$. For brevity reasons, we have omitted the details. In passing, we note that the remaining integral from u_1 to u_2 can be easily computed numerically. In the examples of Section 5, we have used Simpson's "3/8" rule to this end.

Example

Consider a system of two parallel heterogeneous servers with rates $\nu_1 = 1$ and $\nu_2 = 2$, where exponentially distributed jobs with mean size of 1 and constant holding cost of $B = 1$ arrive at rate $\lambda = 2.25$. Thus the offered load is $\rho = 0.75$. As a basic policy, we have RND-ρ, which balances the load between the servers with $p_1 = 1/3$ and $p_2 = 2/3$. Suppose a new job with size $x = 0.5$ arrives when the backlog in Queue 1 is $u'_1 = 2$, while the backlog in Queue 2 is varied. We have two initial actions (Queue 1 or Queue 2) and three second actions (basic policy, LWL and Myopic), which makes it in total six combinations. Fig. 7 illustrates the marginal admission costs for these six actions. The x-axis corresponds to the initial backlog in Queue 2, u'_2, and the y-axis is the marginal cost.

Consider first the basic FPI decisions indicated with "Queue $i + \alpha_0$". This results in a switch-over policy with a switch-over point $\xi_{\text{FPI}} = 2.175$. With LWL as the second action ("Queue i + LWL + α_0"), the marginal costs behave bizarrely at two points, at $u'_2 = 1.75$ and at $u'_2 = 2.5$. At the first point, choosing Queue 2 initially makes the *a posteriori* backlogs equal. Similarly, at the second point with choosing Queue 1 initially. At both points, LWL makes a "mistake" by choosing the queue for the next job solely based on the backlogs and neglecting the service rates, as explained earlier. This is the reason why the marginal cost with LWL can be even higher than without the second action (basic FPI). Moreover, there is some inconsistency as the decision for the given job changes several times between Queue 1 and Queue 2 as backlog u'_2 increases.

In contrast, the dynamic Myopic second action, taking into account also the service rates and the size of the next job, behaves in a robust and coherent manner, yielding constantly the lowest marginal cost (with an appropriate initial queue). In particular, similarly as with the basic FPI, the current job is assigned to Queue 1 when $u'_2 < \xi_{\text{Deep}}$, and otherwise to Queue 2. The difference to the basic FPI is the lower switch-over point, $\xi_{\text{Deep}} \approx 1.8$, whereas $\xi_{\text{FPI}} \approx 2.2$.

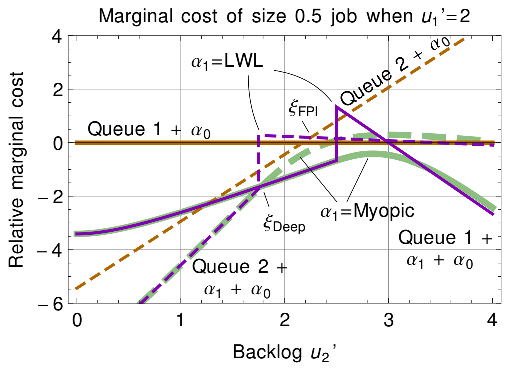


Figure 7: Marginal costs with six different actions for heterogeneous service rates, $\nu_1 = 1$ and $\nu_2 = 2$. Dynamic second action by Myopic has robust behavior unlike LWL, which neglects the service rates.

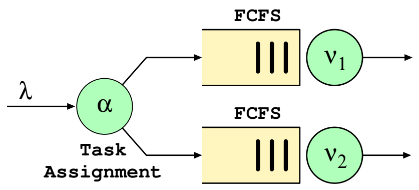


Figure 8: Elementary example system with two parallel queues with service rates ν_1 and ν_2.

5. Numerical experiments

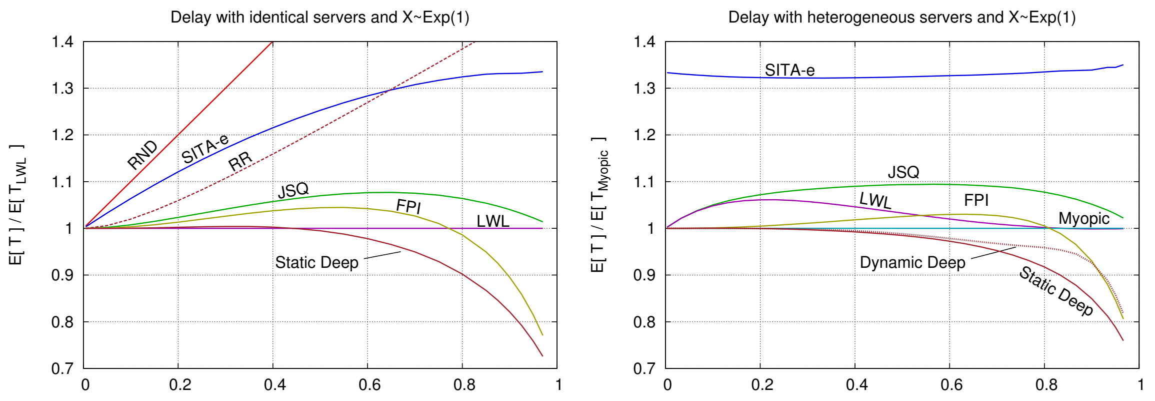
In this section, we consider an elementary system comprising two parallel FCFS servers illustrated in Fig. 8. Our aim is to minimize the mean delay, i.e., $B = 1$. The servers are either identical, $\nu_1 = \nu_2 = 1$, or heterogeneous, $\nu_1 = 1$ and $\nu_2 = 2$. First we assume exponentially distributed job sizes and vary the offered load ρ. Then we assume Weibull and Pareto distributions and experiment with the variability in the job sizes at $\rho = 0.8$. The policies are the following:

1. Bernoulli split (RND-ρ), which balances the load between the servers ($p_i = \nu_i / \sum_j \nu_j$).
2. Round-Robin (RR) in cases where the servers are identical.
3. Join-the-Shortest-Queue (JSQ), which assumes that only the number in queues is available.
4. Least-Work-Left (LWL), choosing the queue with the shortest backlog (ties to faster server).
5. Myopic minimizing the delay of the given job. With identical servers, same as LWL and omitted.
6. Size-Interval-Task-Assignment with equal loads (SITA-e).
7. FPI and Deep (with static/Myopic second action) with α_0=SITA-e.

Note that the above policies require different amounts of information. Bernoulli split and SITA are stateless policies, where the latter requires that jobs' service times are available. Round-robin requires only a minimal state information (the previously chosen queue). In contrast, JSQ requires that the number in queues is available, while LWL, Myopic, FPI and Deep are based on the actual backlogs. Therefore, some policies are not feasible if the required information (e.g., service times) is not available.

5.1. *Exponential distribution*

Let us first assume that the job sizes obey the exponential distribution, $X \sim \text{Exp}(1)$, and we have the two identical servers with service rates $\nu_1 = \nu_2 = 1$. Fig. 9(a) depicts the mean delay as a function of the offered load ρ. Recall that JSQ has been shown to be the optimal policy when only the number in queues is available [5]. Hence, the dynamic size-aware policies should do better, and static Bernoulli-split (RND) worse than JSQ. From the figure we see that indeed LWL, FPI and Deep work well. Initially, FPI has slight problems and is worse than (myopic) LWL. Deep outperforms FPI by a clear margin constantly and it is the single best policy for $\rho > 0.4$. LWL does an equally good job initially, but as the load increases, its greedy decisions start to backfire.



(a) Identical servers (b) Heterogeneous servers

Figure 9: Simulation results with exponentially distributed job sizes. Static Deep is clearly the best policy and outperforms also the Dynamic Deep policy in the heterogeneous setting.

Simulation results with the heterogeneous service rates of $\nu_1 = 1$ and $\nu_2 = 2$ are depicted in Fig. 9(b). Here we have omitted RR as it becomes unstable earlier than other policies. Myopic, which is different from LWL with heterogeneous servers, is used as a reference policy. Note that as $\rho \to 0$, Myopic is (trivially) the optimal policy. As for Deep, we consider both the static (the next job to Queue 1 or Queue 2) and the dynamic (the next job according to Myopic) variants.

FPI turns out to be worse than Myopic until the load increases above 0.8. Static Deep shows a strong performance also in this case and it manages to “slice off” a significant fraction away from the mean delay from $\rho \approx 0.4$ onwards. It is clearly the strongest candidate and, somewhat surprisingly, also better than Dynamic Deep. We tested also Static Deep with α_1=LWL, which evaluates only one second action (i.e., the next job goes to the queue with the shortest backlog). This, however, deteriorated the performance especially in the heterogeneous case, which could have been expected based on Fig. 7.

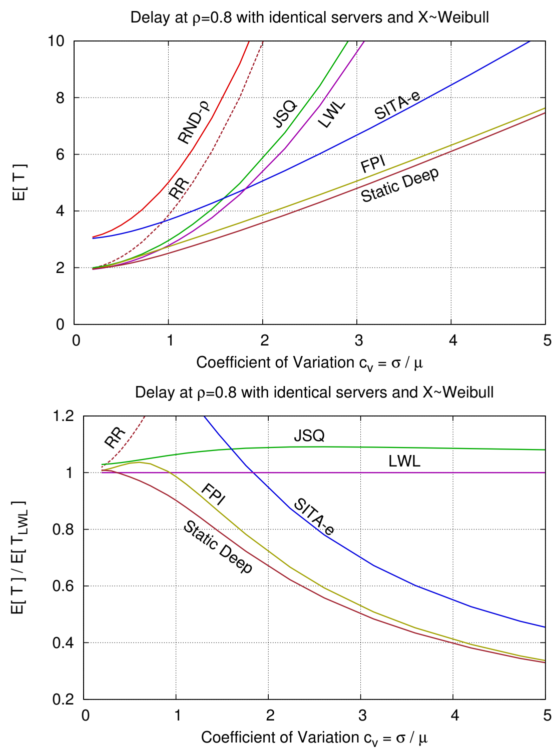
5.2. Weibull distribution

Suppose next that the job sizes obey Weibull distribution, which allows us to adjust the coefficient of variation, c_v, while keeping the mean constant, $\mathrm{E}[X] = 1$. At $c_v = 1$ we obtain the exponential distribution. Then we fix the offered load to a moderately high level, $\rho = 0.8$, where a dispatching policy needs to find an appropriate balance between the short-term (immediate) and long-term costs (queue lengths). Fig. 10 depicts the simulation results with the aforementioned policies as a function of c_v. The upper row depicts the absolute delay, and the lower row illustrates the relative performance. In the left figures, the servers are identical, and in the right figures they are heterogeneous.

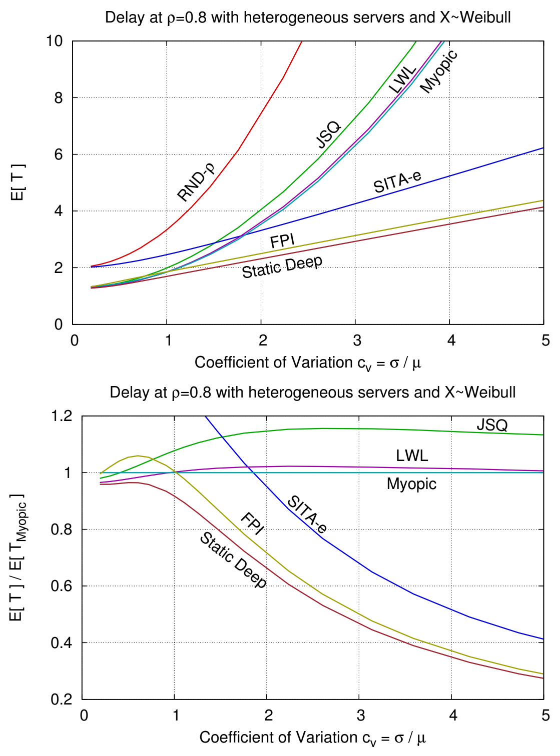
Consider first the identical servers. Initially, the policies fall naturally into two groups, static and dynamic. As c_v increases, we can identify 4 groups: (i) RND-ρ and RR, (ii) JSQ and LWL, (iii) SITA-e, and (iv) FPI and Deep. RND-ρ is the weakest and equal to SITA-e as $c_v \to 0$. RR is initially optimal at $c_v = 0$, but soon loses the edge and follows RND-ρ. JSQ is slightly worse than LWL, which can be explained by the smaller amount of available information. As known from literature, SITA-e passes JSQ and LWL[2] as c_v increases above 1..2. Applying the policy improvement step to SITA-e gives us a clearly improved policy (FPI), which achieves a lower delay than LWL for $c_v > 1$ and stays ahead of SITA-e by a significant margin across all values of c_v. Finally, Deep provides a clear improvement over FPI especially for small values of c_v, where the chosen basic policy, SITA-e, is still weak. As c_v increases, the performance of FPI and Deep converge.

With heterogeneous servers, the situation is very similar. Static Deep turned out to be slightly better than Dynamic Deep (with fixed Myopic α_1) and for clarity reasons only the static variant has been included. We notice that Myopic

[2]It is also good to remind ourselves about the findings by Harchol-Balter *et al.* in [25], where it is shown that SITA can actually be worse than LWL for some distributions when c_v is sufficiently large.



(a) Identical servers



(b) Heterogeneous servers

Figure 10: Mean delay (i.e. latency) with Weibull-distributed job sizes.

is marginally better than LWL for larger values of c_v. Static Deep is again the strongest and outperforms FPI by a higher margin initially, but the difference becomes smaller as c_v increases.

5.3. Pareto distribution

Suppose next that the job sizes obey Pareto distribution, $X \sim \text{Pareto}(k, 10000, \alpha)$, i.e., a truncated Pareto distribution to $(k, 10000)$ with shape parameter α. Again, we scale the mean to one and vary the coefficient of variation c_v by adjusting k and α appropriately. The offered load is set to $\rho = 0.8$.

The simulation results are depicted in Fig. 11. Deep is superior again with both identical and heterogeneous servers. In contrast to exponential and Weibull distribution, in this case Dynamic Deep turns out to be a slightly better option than Static Deep with heterogeneous servers, and the latter is omitted for clarity. The improvement by FPI from SITA-e is clearly the most significant (as usual). Deep then makes smaller adjustments to the policy, and manages still to “slice off” a considerable fraction away from the mean delay. We believe that the decisions of Deep indeed are near optimal (in these cases).

6. Conclusions

The proposed lookahead approach enables a finer control over the actions in the near future, and (partially) overcomes the obstacles that arise when analyzing dynamic dispatching policies in the MDP framework. In particular, the lookahead policy improvement approach (Deep in short) makes it possible to mimic dynamic dispatching policies such as LWL and Myopic, yielding sound estimates for the corresponding marginal costs. Conceptually, we seek lower costs by considering also the subsequent decisions that apply tentatively to the later arriving jobs. That is, we consider decisions such as choosing Queue i for the current job and Queue j for the next, where the specifics of the latter job are currently unknown. The lookahead step is important as the benefits from some actions can only be reaped with appropriate subsequent actions. In the numerical experiments, we observed significant improvements in

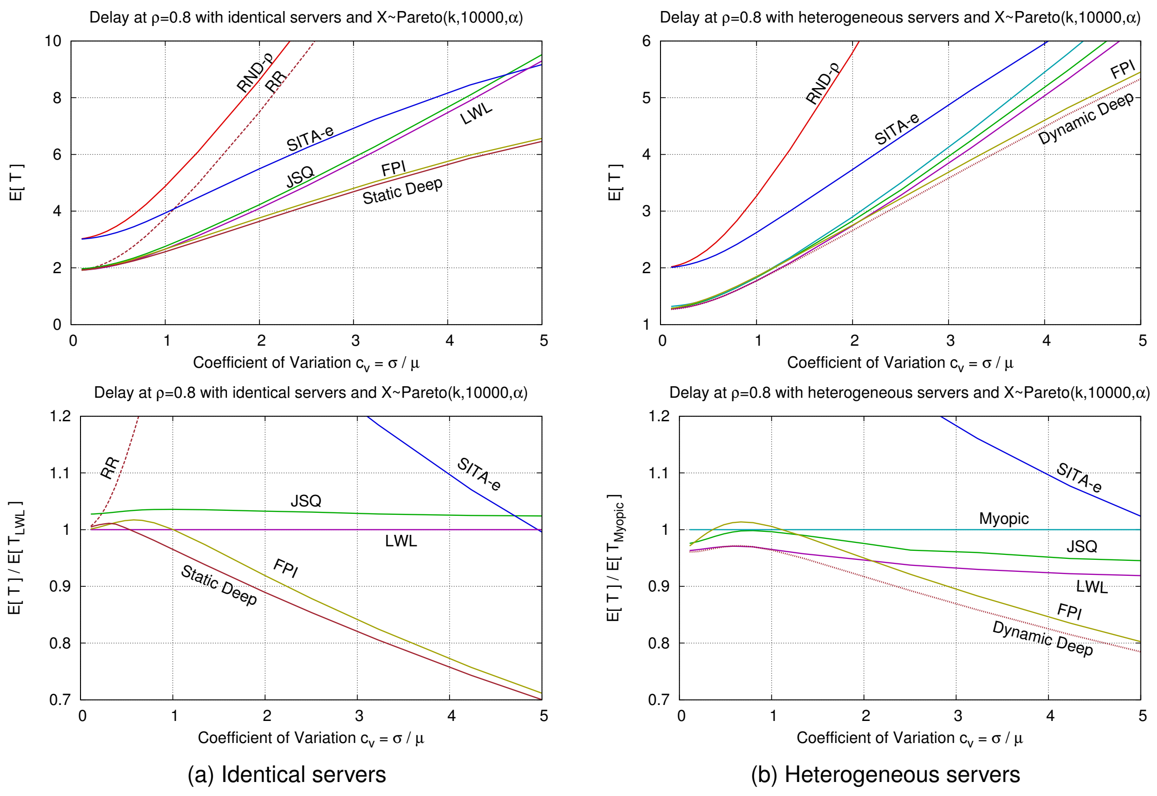


Figure 11: Performance of different policies with a truncated Pareto distribution.

performance over the existing dispatching policies in many cases. When improvements were smaller, the reference policies are assumed to be already close to optimal (e.g., LWL is known to be optimal with identical servers and fixed job sizes). Furthermore, we only experimented with the mean delay, while the gain from FPI and Deep in particular can be expected to be higher with more diverse cost structures (by non-constant holding costs).

With static decisions, it seems to be straightforward to go beyond the two actions, e.g., by expanding the lookahead to three steps. For example, a decision for the third job according to LWL can be easily deduced by conditioning on the size of the second job. Alternatively, one can evaluate all triplets (i_0, i_1, i_2), or apply some pruning if the number of queues is too large. Unfortunately, with dynamic actions the combinatorial complexity appears to become a formidable challenge immediately and more advanced techniques and/or alternative approaches should be considered.

Appendix: Alternative offset for marginal costs

Let us assume a constant holding cost, $B = 1$. The marginal admission cost in general is

$$c = h + v_{\mathbf{z}} - v_{\mathbf{z}'},$$

i.e., the sum of the immediate cost h and the increase in the value function. As the initial state $\mathbf{z}'$ is common to all decisions, we can also add $v_{\mathbf{z}'} - v_0$ to above. Then, for the basic FPI, the (relative) marginal cost is

$$c_i(x) = u_i + \sum_i g_i u_i^2.$$

Similarly, for the static (i_0, i_1) action,

$$c_{i_0,i_1}(x) = u_{i_0} - \mathrm{E}[T] + \frac{1}{\lambda^2}\sum_i g_i(2 + \lambda u_i(\lambda u_i - 2) - 2e^{-\lambda u_i}) + \left(1+\frac{2g_{i_1}\,\mathrm{E}[X]}{\nu_{i_1}}\right)\left(u_{i_1}-\frac{1-e^{-\lambda u_{i_1}}}{\lambda}\right) + g_{i_1}\frac{\mathrm{E}[X^2]}{\nu_{i_1}^2}+\frac{\mathrm{E}[X]}{\nu_{i_1}},$$

where the last two terms correspond to the mean delay in Queue i_1 if α_0 is a Bernoulli split. For the dynamic second action we obtain

$$\begin{aligned} c_{i_0,\alpha_1}(x) = u_{i_0} - \mathrm{E}[T] + \frac{1}{\lambda^2} \sum_i g_i(2+\lambda u_i(\lambda u_i-2)-2e^{-\lambda u_i}) \\ + \sum_{i=1}^{k} \int_0^\infty \lambda e^{-\lambda t}\, \mathrm{P}\{X \in I_i(t)\} \Big((u_i-t)^+(1+2g_i\mathrm{E}[S_i(t)])+g_i\mathrm{E}[S_i(t)^2] + \mathrm{E}[S_i(t)] \Big)\, dt. \end{aligned}$$

These equations are somewhat more compact as common terms have been subtracted away and $B = 1$ is assumed.

References

[1] O. Akgun, R. Righter, R. Wolff, Understanding the marginal impact of customer flexibility, Queueing Systems 71 (1-2) (2012) 5–23.

[2] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, M. S. Squillante, Analysis of task assignment with cycle stealing under central queue, in: Proc. of the 23rd International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, 2003, pp. 628–637.

[3] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, M. S. Squillante, Cycle stealing under immediate dispatch task assignment, in: Proc. of the fifteenth annual ACM symposium on Parallel algorithms and architectures (SPAA), ACM, New York, NY, USA, 2003, pp. 274–285.

[4] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, 2013.

[5] W. Winston, Optimality of the shortest line discipline, Journal of Applied Probability 14 (1977) 181–189.

[6] A. Ephremides, P. Varaiya, J. Walrand, A simple dynamic routing problem, IEEE Transactions on Automatic Control 25 (4) (1980) 690–693.

[7] Z. Liu, D. Towsley, Optimality of the round-robin routing policy, Journal of Applied Probability 31 (2) (1994) 466–475.

[8] Z. Liu, R. Righter, Optimal load balancing on distributed homogeneous unreliable processors, Operations Research 46 (4) (1998) 563–573.

[9] M. E. Crovella, M. Harchol-Balter, C. D. Murta, Task assignment in a distributed system: Improving performance by unbalancing load, in: Proceedings of SIGMETRICS '98, Madison, Wisconsin, USA, 1998, pp. 268–269.

[10] M. Harchol-Balter, M. E. Crovella, C. D. Murta, On choosing a task assignment policy for a distributed server system, Journal of Parallel and Distributed Computing 59 (1999) 204–228.

[11] H. Feng, V. Misra, D. Rubenstein, Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems, Perform. Eval. 62 (1-4) (2005) 475–492.

[12] R. Bellman, A Markovian decision process, Indiana Univ. Math. J. 6 (1957) 679–684.

[13] R. A. Howard, Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes, Wiley Interscience, 1971.

[14] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, 2005.

[15] K. R. Krishnan, T. J. Ott, State-dependent routing for telephone traffic: Theory and results, in: IEEE Conference on Decision and Control, Vol. 25, 1986, pp. 2124–2128.

[16] K. R. Krishnan, Markov decision algorithms for dynamic routing, IEEE Communications Magazine (1990) 66–69.

[17] J. van Leeuwaarden, S. Aalto, J. Virtamo, Load balancing in cellular networks using first policy iteration, Technical report, Networking Laboratory, Helsinki University of Technology (2001).

[18] K. R. Krishnan, Joining the right queue: a state-dependent decision rule, IEEE Transactions on Automatic Control 35 (1) (1990) 104–108.

[19] S. A. E. Sassen, H. C. Tijms, R. D. Nobel, A heuristic rule for routing customers to parallel servers, Statistica Neerlandica 51 (1) (1997) 107–121.

[20] S. Bhulai, On the value function of the M/Cox(r)/1 queue, Journal of Applied Probability 43 (2) (2006) 363–376.

[21] E. Hyytiä, J. Virtamo, S. Aalto, A. Penttinen, M/M/1-PS queue and size-aware task assignment, Perform. Eval. 68 (11) (2011) 1136–1148.

[22] E. Hyytiä, A. Penttinen, S. Aalto, Size- and state-aware dispatching problem with queue-specific job sizes, European Journal of Operational Research 217 (2) (2012) 357–370.

[23] E. Hyytiä, S. Aalto, A. Penttinen, Minimizing slowdown in heterogeneous size-aware dispatching systems, ACM SIGMETRICS Performance Evaluation Review 40 (2012) 29–40, (ACM SIGMETRICS/Performance conference).

[24] M. Harchol-Balter, K. Sigman, A. Wierman, Asymptotic convergence of scheduling policies with respect to slowdown, Perform. Eval. 49 (1-4) (2002) 241–256.

[25] M. Harchol-Balter, A. Scheller-Wolf, A. R. Young, Surprising results on task assignment in server farms with high-variability workloads, in: Proc. of SIGMETRICS, ACM, New York, NY, USA, 2009, pp. 287–298.